Reoptimization of the Maximum Weighted P k -Free Subgraph Problem under Vertex Insertion

Reoptimization of the Maximum Weighted P_k-Free Subgraph Problem under Vertex Insertion*

Nicolas Boria[1], Jérôme Monnot[1], and Vangelis Th. Paschos[1,2]

[1] LAMSADE, CNRS UMR 7243 and Université Paris-Dauphine
{boria,monnot,paschos}@lamsade.dauphine.fr
[2] Institut Universitaire de France

Abstract. The reoptimization issue studied in this paper can be described as follows: given an instance I of some problem Π, an optimal solution OPT for Π in I and an instance I' resulting from a local perturbation of I that consists of insertions or removals of a small number of data, we wish to use OPT in order to solve Π in I', either optimally or by guaranteeing an approximation ratio better than that guaranteed by an ex nihilo computation and with running time better than that needed for such a computation. In this setting we study the weighted version of MAX WEIGHTED P_k-FREE SUBGRAPH. We then show, how the technique we use allows us to handle also BIN PACKING.

1 Introduction

Hereditary problems in graphs, also known as maximal subgraph problems, include a wide range of classical combinatorial optimization problems, such as MAX WEIGHTED INDEPENDENT SET or MAX WEIGHTED H-FREE SUBGRAPH. Most of these problems are known to be **NP**-hard, and even inapproximable within any constant approximation ratio unless $\mathbf{P} = \mathbf{NP}$ [16, 17].

In what follows, we study approximation of such a problem, namely the maximum weight P_k-FREE SUBGRAPH, denoted by MAX WEIGHTED P_k-FREE SUBGRAPH, in the reoptimization setting, which can be described as follows: considering an instance I of a given problem Π with a known optimum OPT, and an instance I' which results from a local perturbation of I, can the information provided by OPT be used to solve I' in a more efficient way (i.e., with a lower complexity and/or with a better approximation ratio) than if this information wasn't available?

The reoptimization setting was introduced in [1] for METRIC TSP. Since then, many other optimization problems were discussed in this setting, including STEINER TREE [5, 9, 10, 15], MINIMUM SPANNING TREE [14], as well as various versions of TSP [4, 8, 12]. In all cases, the goal is to propose reoptimization algorithm that outperform their deterministic counterparts in terms of complexity

*Research supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010

and/or approximation ratio. In [7], the MAX WEIGHTED INDEPENDENT SET problem, as well as MIN WEIGHTED VERTEX COVER and MIN WEIGHTED SET COVER problems, are discussed in a similar setting up to the fact that perturbations there concerned the edge-set of the initial graph. The authors of [7] manage to provide optimal approximation results (i.e., upper and lower bounds matching ones the others) under the basic assumption that the initial solution is not necessarily optimal but ρ-approximate. Finally, let us note that in [6] reoptimization variants of the shortest common superstring problem are considered, where the local modifications consist of adding or removing a single string.

When one deals with hereditary problems, and I' results from a perturbation of the vertex set (insertion or deletion), solutions of I remain feasible in I'. This property is very interesting when reoptimizing hereditary problems, and makes most of them APX in the reoptimization setting. For exemple, a very simple algorithm provides a $(1/2)$-approximation for a whole class of hereditary problems, including MAX WEIGHTED INDEPENDENT SET when a single vertex is inserted [3]. Let us note that the unweighted versions of all these problems in this setting admit polynomial time approximation schemata. In what follows, we improve on this result by presenting algorithms designed for two specific hereditary problems, and also provide inapproximability bounds.

The paper is organized as follows: general properties regarding reoptimization and the MAX WEIGHTED P_k-FREE SUBGRAPH are presented in what follows in this section, while in Section 2 we present approximation and inapproximability results under vertex insertion for MAX WEIGHTED P_k-FREE SUBGRAPH. Our results are optimal (in the sense given two paragraphs above). To the best of our knowledge no such results exist in the reoptimization literature for the vertex insertion setting. In Section 3, the BIN PACKING problem is handled.

This paper is part of a larger work [13] devoted to the study of five maximum weight induced hereditary subgraph problems, namely, MAX WEIGHTED INDEPENDENT SET, MAX WEIGHTED k-COLORABLE SUBGRAPH, MAX WEIGHTED P_k-FREE SUBGRAPH, MAX WEIGHTED SPLIT SUBGRAPH and MAX PLANAR SUBGRAPH. For reasons of length limits some of the results are given without detailed proofs, the interested reader is referred to [13] where all proofs can be found.

Before presenting properties and results regarding reoptimization problems, we will first give formal definitions of what are reoptimization problems, reoptimization instances, and approximate reoptimization algorithms:

Definition 1. *An optimization problem Π is defined by a quadruple:*

$$(\mathcal{I}_\Pi, Sol_\Pi, m_\Pi, goal(\Pi))$$

where $\mathcal{I}_\Pi$ is the set of instances of Π; given $I \in \mathcal{I}_\Pi$, $Sol_\Pi(I)$ is the set of feasible solutions of I; given $I \in \mathcal{I}_\Pi$, and $S \in Sol_\Pi(I)$, $m_\Pi(I, S)$ denotes the value of the solution S of the instance I; $goal(\Pi) \in \{\min, \max\}$.

A reoptimization problem $R\Pi$ is given by a pair $(\Pi, R_{R\Pi})$ where: Π is an optimization problem as defined in Definition 1; $R_{R\Pi}$ is a rule of modification on instances of Π, such as addition, deletion or alteration of a given amount of data;

given $I \in \mathcal{I}_{\Pi}$ and $R_{R\Pi}$, $modif_{R\Pi}(I, R_{R\Pi})$ denotes the set of instances resulting from applying modification $R_{R\Pi}$ to I; notice that $modif_{R\Pi}(I, R_{R\Pi}) \subset \mathcal{I}_{\Pi}$.

For a given reoptimization problem $R\Pi(\Pi, R_{R\Pi})$, a reoptimization instance $I_{R\Pi}$ of $R\Pi$ is given by a triple (I, S, I'), where: I denotes an instance of Π, referred to as the *initial* instance; S denotes a feasible solution for Π on the initial instance I; I' denotes an instance of Π in $modif_{R\Pi}(I, R_{R\Pi})$; I' is referred to as the *perturbed* instance. For a given instance $I_{R\Pi}(I, S, I')$ of $R\Pi$, the set of feasible solutions is $Sol_{\Pi}(I')$.

Definition 2. *For a given reoptimization problem $R\Pi(\Pi, R_{R\Pi})$, a reoptimization algorithm $\mathtt{A}$ is said to be a ρ-approximation reoptimization algorithm for $R\Pi$ if and only if: (i) $\mathtt{A}$ returns a feasible solution on all instances $I_{R\Pi}(I, S, I')$; (ii) $\mathtt{A}$ returns a ρ-approximate solution on all reoptimization instances $I_{R\Pi}(I, S, I')$ where S is an optimal solution for I.*

Note that Definition 2 is the most classical definition found in the literature, as well as the one used in this paper. However, an alternate (and more general) definition exists (used for example in [5, 7, 9, 10]), where a ρ_1-approximation reoptimization algorithm for $R\Pi$ is supposed to ensure a $\rho_1\rho_2$-approximation on any reoptimization instance $I_{R\Pi}(I, S, I')$ where S is a ρ_2-approximate solution in the initial instance I.

A property $\mathcal{P}$ on a graph is hereditary if the following holds: if the graph satisfies $\mathcal{P}$, then $\mathcal{P}$ is also satisfied by all its induced subgraphs. Following this definition, independence, planarity, bipartiteness are three examples of hereditary properties: in a given graph, any subset of an independent set is an independent set itself, and the same holds for planar and bipartite subgraphs. On the opposite hand, connectivity is no hereditary property since there might exist some subsets of G whose removal disconnect the graph.

We denote by hereditary problem any problem that consists of finding the maximum set of vertices (in terms of cardinality or weight) that induces a subgraph verifying a given hereditary property.

A graph is said to be P_k-free if it does not contain a path on k edges. Here, P_k-free means that the graph does not admit a P_k as *minor* (and not as *induced subgraph*). A graph H is a minor of a graph G, if H can be obtained by a sequence of edge contractions and vertex or edge deletions. For example, a C_{k+1} admits a P_k as minor, so, although it does not admit a P_k as induced subgraph, we consider that a C_{k+1} is not P_k-free.

Let $G(V, E, w)$ be a vertex-weighted graph with $w(v) \geqslant 0$, for any $v \in V$. The MAX WEIGHTED P_k-FREE SUBGRAPH problem is the problem consisting, given a graph $G(V, E, w)$, of finding a subset S of vertices such that $G[S]$ is P_k-free and maximizes $w(S) = \sum_{v \in S} w(v)$. For instance, MAX WEIGHTED INDEPENDENT SET is exactly the MAX WEIGHTED P_2-FREE SUBGRAPH problem, while MAX WEIGHTED P_3-FREE SUBGRAPH consists of finding an independent set and an induced matching of maximum total weight.

As it is proved in [16] (see Theorem 1 just below) most hereditary problems (hence MAX WEIGHTED P_k-FREE SUBGRAPH also) are highly inapproximable unless $\mathbf{P} = \mathbf{NP}$.

Theorem 1. ([16]) *There exists an* $\varepsilon \in (0,1)$ *such that* MAX WEIGHTED P_k-FREE SUBGRAPH *cannot be approximated with ratio* $n^{-\varepsilon}$ *in polynomial time unless* $\mathbf{P} = \mathbf{NP}$.

In the sequel, G_p and G'_p will denote initial and perturbed instances, while OPT_p and OPT'_p will denote optimal solutions in G_p and G'_p, respectively. For simplicity and when no confusion arises, we will omit subscript p. The function w refers to the weight function, taking a vertex, a vertex set, or a graph as input (the weight of a graph is defined as the sum of weights of its vertices). Finally, note that throughout the whole paper, the term "subgraph" will always implicitly refer to "induced subgraph".

Under vertex insertion, the inapproximability bounds of Theorem 1 is easily broken. In [3], a very simple strategy, denoted by `R1` in what follows, provides a $(1/2)$-approximation for any hereditary problem. This strategy consists of outputting the best solution among the newly inserted vertex and the initial optimum. Moreover, this strategy can also be applied when a constant number h of vertices is inserted: it suffices to output the best solution between an optimum in the h newly inserted vertices (that can be found in $O(2^h)$ through exhaustive search) and the initial optimum. The $1/2$ approximation ratio is also ensured in this case [3].

Note that an algorithm similar to `R1` was proposed for KNAPSACK in [2]. Indeed, this problem, although not being a graph problem, it is hereditary in the sense defined above, so that returning the best solution between a newly inserted item and the initial optimum ensures a $(1/2)$-approximation ratio. The authors also show that any reoptimization algorithm that does not consider objects discarded by the initial optimal solution cannot have ratio better than $1/2$.

2 MAX WEIGHTED P_k-FREE SUBGRAPH

The MAX WEIGHTED P_k-FREE SUBGRAPH problem discussed in this subsection refers to MAX WEIGHTED P_k-FREE SUBGRAPH (with P_k as forbidden minor), and not to MAX WEIGHTED INDUCED P_k-FREE SUBGRAPH (with P_k as forbidden induced subgraph). Formally, given a graph $G(V, E, w)$ and a constant $k \leqslant n$, the MAX WEIGHTED P_k-FREE SUBGRAPH problem handled in this section consists of finding a maximum-total weight set of vertices that induces a subgraph of G that is P_k-free. The approximability analysis of MAX WEIGHTED P_k-FREE SUBGRAPH uses the following lemma whose proof can be found in [13].

Lemma 1. *A* P_k*-free graph can be colored with* k *colors in polynomial time.*

Proposition 1. *Under one vertex insertion,* MAX WEIGHTED P_k-FREE SUBGRAPH *is inapproximable within ratio* $\frac{2k}{3k+1} + \varepsilon$ *in polynomial time if* k *is odd, and inapproximable within ratio* $\frac{2k}{3k+2} + \varepsilon$ *if* k *is even, unless* $\mathbf{P} = \mathbf{NP}$.

Proof (Sketch). The technique used for the proof can be sketched as follows. Considering an unweighted graph $H(V, E)$ on which one wants to solve a given

hereditary problem Π, known to be inapproximable within any constant ratio, we build a reoptimization instance I_p, where p denotes a vector of fixed size (i.e., independent of the size n of G; so, $|p|$ is a fixed constant) that contains integer parameters between 1 and n. This instance is characterized by an initial graph G_p (that contains H), with a known solution, and a perturbed instance G'_p.

Then, we prove that, for some specific (yet unknown) value p' of the parameter vector p (that is, when the value of p coincides with some unknown structural parameters of the graph, like independence number for example), an optimal solution can be easily determined in the initial graph $G_{p'}$, and a ρ-approximate solution $S_{p'}$ in $G'_{p'}$ necessarily induces a solution $S_{p'}[V]$ in H, that is a constant approximation for the initial problem. Considering that the vector p can take at most $n^{|p|}$ possible values, it is possible in polynomial time to build all instances I_p, to run the polynomial ρ-approximation algorithm on all of them, and to return the best set $S_{p^*}[V]$ as solution for Π in H. The whole procedure is polynomial and ensures a constant-approximation for Π, which is impossible unless $\mathbf{P} = \mathbf{NP}$, so that a ρ-approximation algorithm cannot exist for the considered reoptimization version of Π, unless $\mathbf{P} = \mathbf{NP}$.

Consider now a connected graph H that has independence number α, on which one wishes to solve (or approximate) MAX WEIGHTED INDEPENDENT SET problem. Transform H into a graph $H_k(V, E)$ in the following way: each vertex v_i of H is turned into a clique V_i of k vertices in H_k; if (v_i, v_j) belongs to H, then all edges between vertices of cliques V_i and V_j are in E. Note that H_k is connected as H itself is also connected. Considering that the biggest independent set in H has size α, then the same holds in H_k: an independent set in G can only take one vertex in each clique V_i, and it cannot take two vertices from cliques that corresponds to neighbors in H.

Following Lemma 1, any P_k-free subgraph in H_k can be partitioned in k independent sets, so that, denoting by OPT an optimal P_k-free subgraph in H_k, $|\text{OPT}| \leqslant k\alpha$. Moreover, denoting by IS an optimal independent set in H (that has exactly α vertices), then, in H_k, the union of all cliques corresponding to vertices of IS induces a P_k-free subgraph with value exactly $k\alpha$, so that $|\text{OPT}| = k\alpha$.

Remark 1. Following the same arguments, it holds that for any $i \leqslant k$ a P_i-free subgraph in H_k cannot have weight more than $i\alpha$ (and an optimal one has weight exactly $i\alpha$).

First, suppose that k is odd. We build a weighted reoptimization instance $I_{\alpha,k}$ of MAX WEIGHTED P_k-FREE SUBGRAPH as follows (Figure 1 provides a representation of the general structure).

– The initial graph $G_{\alpha,k}$ is obtained by adding to the graph H_k a set X of vertices which consists of two cliques X_1, and X_2 to G. Both these cliques have k vertices, each with weight $k\alpha$. Clique X_1, is divided into two subcliques X_{1C} and X_{1NC}; X_{1C} has $(k+1)/2$ vertices that are all connected to all vertices in V, while the other $(k-1)/2$ vertices of X_{1NC} are not connected to any vertex in V. Clique X_2 is divided in the same manner. Finally, each vertex in V receives

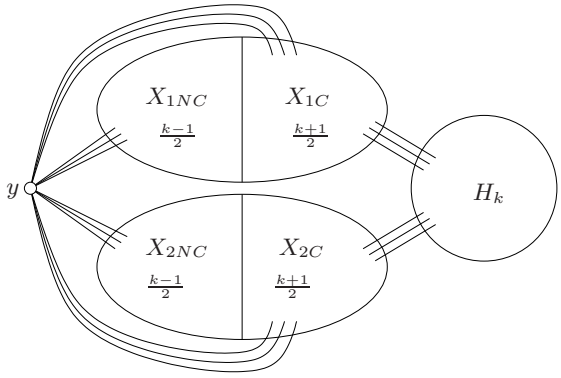

Fig. 1. Reoptimization instance $I_{\alpha,k}$

weight $k+1$, hence, for any $1 \leqslant i \leqslant k$ an optimal P_i-free subgraph inside V has weight exactly $i(k+1)\alpha$.

– The perturbed graph $G'_{\alpha,k}$ is obtained by adding a single vertex y to G_α, with weight $k(k+1)\alpha$, that is connected to all vertices of X.

We first prove that X defines an optimum on the initial graph $G_{\alpha,k}$, so that the reoptimization instance $I_{\alpha,k}$ is well defined. Then, we show that $\{y\} \cup X_{1NC} \cup X_{2NC} \cup F^*$ (where F^* is an optimum in H_k) is feasible in the graph $G'_{\alpha,k}$, so that $w(\mathrm{OPT}') \geqslant w(\{y\} \cup X_{1NC} \cup X_{2NC} \cup F^*) = k(3k+1)\alpha$. This lower bound on the weight of the optimum, naturally induces a lower bound on the weight of a $\left(\frac{2k}{3k+1} + \varepsilon\right)$-approximate solution, say $S_{\alpha,k}$: $w\left(S_{\alpha,k}\right) \geqslant \left(\frac{2k}{3k+1} + \varepsilon\right) w(\mathrm{OPT}') \geqslant (1+\varepsilon)2k^2\alpha$.

We then show that any feasible solution on the subgraph induced by $\{y\} \cup X$ cannot have weight more than $2k^2\alpha$, and this holds a fortiori for the restriction of $S_{\alpha,k}$ to this subgraph, denoted by $S_{\alpha,k}[\{y\} \cup X]$. Combining this fact with the expression for $w(S_{\alpha,k})$ above, and taking into account that $\{\{y\} \cup X, V\}$ defines a partition of the final graph, we derive that $w(S_{\alpha,k}[V]) = w(S_{\alpha,k}) - w(S_{\alpha,k}[\{y\} \cup X]) \geqslant 2\varepsilon k^2\alpha$. Hence, a $\left(\frac{2k}{3k+1} + \varepsilon\right)$-approximation algorithm for the reoptimization version of MAX WEIGHTED P_k-FREE SUBGRAPH can be used to derive a polynomial ε-approximation algorithm for the static version of MAX WEIGHTED INDEPENDENT SET, which is impossible unless $\mathbf{P} = \mathbf{NP}$.

In the same way, we present a $\left(\frac{2k}{3k+2} + \varepsilon\right)$ inapproximability bound for the case where k is even. The structure of the instance is somewhat more complex, but the structure of the proof itself is similar. A full proof of both inapproximability bounds can be found in [13].

Remark 2. The proof of Proposition 1 works also even if we assume that a ρ-approximate solution is given instead of an optimal one. In this case, the bounds claimed in Proposition 1 are simply multiplied by ρ.

We now prove that ratio of Proposition 1 is tight for small values of k, namely when $k \leqslant 6$. Consider the following hypothesis.

Hypothesis 1. In polynomial time, a P_k-free subgraph S can be partitioned in 3 sets S_1, S_2 and S_3, such that both S_1 and S_2 are $P_{\lceil k/2 \rceil - 1}$-free, and $w(S_3) \leqslant w(S)/k$ if k is odd and $w(S_3) \leqslant 2w(S)/k$ if k is even. ∎

Proposition 2. *For values of k for which Hypothesis 1 is true,* MAX WEIGHTED P_k-FREE SUBGRAPH *problem is approximable within ratio $\frac{2k}{3k+1}$ for odd values of k, and $\frac{2k}{3k+2}$ for even values of k in the reoptimization setting, under one vertex insertion.*

Proof. Consider a reoptimization instance I of MAX WEIGHTED P_k-FREE SUBGRAPH, given by two graphs G (initial) and G' (perturbed) and an optimal solution OPT on the initial graph G. Graphs G and G' differ only by vertex y (and its incident edges) which belongs to G' but not to G. Classically, denoting by OPT$'$ an optimal solution on G', it holds that $w(\text{OPT}) \geqslant w(\text{OPT}') - w(y)$. Suppose that Hypothesis 1 is verified, and consider the following algorithm:
partition OPT *in 3 sets S_1, S_2 and S_3 as defined in Hypothesis 1, and without loss of generality, suppose $w(S_1) \geqslant w(S_2)$; set* $\text{SOL}_1 = S_1 \cup \{y\}$ *and* $\text{SOL}_2 = \text{OPT}$*; return the best solution* SOL *between* SOL_1 *and* SOL_2.

First, let us prove that this algorithm returns a feasible solution: SOL_2 is trivially feasible in G', and consider a path P in SOL_1. If this path does not go through y then it cannot go through more than $\lceil k/2 \rceil - 1$ vertices (by hypothesis, S_1 is $P_{\lceil k/2 \rceil - 1}$-free). If it does go through y, then denote by P_1 the set of vertices visited before y in P and P_2 the set of vertices visited after. Considering that both P_1 and P_2 are included in S_1, which is supposed to be $P_{\lceil k/2 \rceil - 1}$-free, then both $|P_1|, |P_2| \leqslant \lceil k/2 \rceil - 1$, so that $|P| = |P_1| + |P_2| + 1 \leqslant k$, and thus SOL_1 is also P_k-free.

Let r_k be an integer that is equal to 1 if k is odd, and to 2 if k is even. Regarding the partitioning induced by Hypothesis 1, it holds that $w(S_3) \leqslant \frac{r_k w(\text{OPT})}{k}$, and thus $w(S_1) \geqslant (w(\text{OPT}) - w(S_3))/2 \geqslant \frac{k - r_k}{2k} w(\text{OPT})$, thus $w(\text{SOL}_1) \geqslant \frac{k-r_k}{2k} w(\text{OPT}) + w(y) = \frac{k-r_k}{2k} w(\text{OPT}') + \frac{k+r_k}{2k} w(y)$.

On the other hand, $w(\text{SOL}_2) \geqslant w(\text{OPT}') - w(y)$. Summing expressions for $w(\text{SOL}_1)$ and $w(\text{SOL}_2)$ with coefficients 1 and $\frac{k+r_k}{2k}$ respectively, one finally proves that $\frac{3k+r_k}{2k} w(\text{SOL}) \geqslant w(\text{SOL}_2) + \frac{k+r_k}{2k} w(\text{SOL}_1) \geqslant w(\text{OPT}')$.

Note once more that, as it can be seen from the proof of Proposition 2, Remark 2 always holds.

Proposition 3. *Hypothesis 1 holds for $k \leqslant 6$.*

Proof (Sketch). In what follows, we suppose that the P_k-free graph S to be partitioned is connected (if it is not so, then proving that the hypothesis is true for each of its connected components amounts to proving it for the whole graph).

$\mathbf{k = 1, 2}$. Simply set $S_3 = S$ and $S_1, S_2 = \emptyset$.

k = 3. Split the graph in three independent sets S'_1, S'_2,and S'_3 in polynomial time (that is possible according to Lemma 1), and w.l.o.g., suppose $w(S'_1) \geqslant w(S'_2) \geqslant w(S'_3)$. Finally, set $S_1 = S'_1$, $S_2 = S'_2$, and $S_3 = S'_3$.

k = 4. Split the graph in four independent sets from S'_1 to S'_4 in polynomial time, and w.l.o.g., suppose $w(S'_1) \geqslant w(S'_2) \geqslant w(S'_3) \geqslant w(S'_4)$. Finally, set $S_1 = S'_1$, $S_2 = S'_2$, and $S_3 = S'_3 \cup S'_4$.

k = 5. This case requires a rather long and involved analysis, which could not be included in the paper due to length limits. A full proof can be found in [13].

k = 6. In this case, Hypothesis 1 is verified if S can be partitioned in 3 P_2-free subgraphs (the lightest of which will be S_3, and the other two S_1 and S_2). Here, we distinguish the following four cases.

Case 1. S contains a C_6. It is clear that S contains exactly 6 vertices, which can easily be partitioned in three P_2-free subgraphs (2 vertices in each set of the partition).

Case 2. S contains a C_5 and no C_6. Let S' denote the set of vertices that are not in C_5. It holds that S' is an independent set, and that a pair of vertices that are neighbors in the C_5 cannot both have neighbors in S' (if one of these two properties is not verified, then there exists a P_6 in S. Thus, without making assumption on the number of chords in C_5, the general structure of S is as described in Figure 2, with a partitioning in three P_2-free subgraphs (represented in the figure as white, grey, and black vertices).

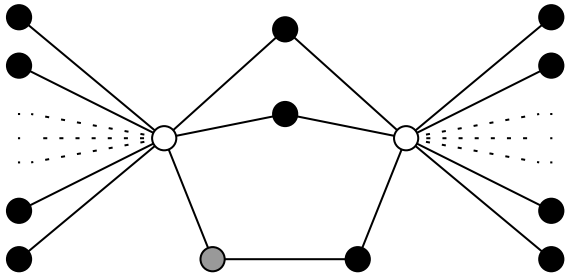

Fig. 2. Partition of P_6-free graph in 3 P_2-free subgraphs in Case 2

Case 3. S contains a C_4. We distinguish here the following two subcases.

Case 3-a. S contains a C_4 but no C_5 and no C_6, and denoting by S' the set of vertices that are not in the C_4, S' contains at least an edge (v_1, v_2). Obviously, these two vertices v_1 and v_2 cannot be connected to two different vertices of the C_4, otherwise, S would contain a C_5 or a C_6.

Moreover, at least one of these two vertices must be connected to a vertex of the C_4. Indeed, taking into account that S is supposed to be connected, if none of these vertices is connected to a vertex of the C_4, then there exists a P_6 in S.

Finally, supposing w.l.o.g that v_1 is the vertex connected to a vertex x in the C_4, then no neighbor of x in the C_4 can be connected to any vertex of S' (otherwise there would exist a P_6 in S).

Hence, denoting vertices of the C_4 by x, y, z and t as in Figure 3, only vertices x and z might have neighbors in S'. Moreover, denoting by $N(\{x\}\cup\{z\})$ the neighborhood of these two vertices, it holds that this set is P_2 free, otherwise a path on 7 vertices would exist in S (the 3 vertices of the P_2 in $N(\{x\} \cup \{z\})$, and the 4 vertices of the C_4). Thus, coloring in white the vertices of $N(\{x\}\cup\{z\})$ as well as y and t, and in black all the other vertices of S, one gets a partition of S into two P_2 free colors.

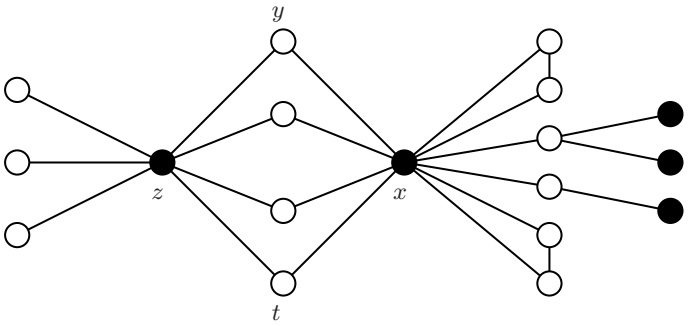

Fig. 3. Partition of P_6-free graph in 2 P_2-free subgraphs in Case 3-a

Case 3-b. S contains a C_4 but no C_5 and no C_6, and let S' denote the set of vertices that are not in the C_4, S' contains no edge. This case is much simpler than the previous one, considering that S' forms an independent set. To get a partition of S into three P_2-free subgraphs, it suffices to consider S' itself as the first subset of the partition, $\{x\} \cup \{z\}$ as the second, and $\{y\} \cup \{t\}$ as the third.

Case 4. S contains no C_4, no C_5 and no C_6. In this case, it holds that the neighborhood $N(x)$ of any vertex x is P_2-free, since a P_2 (on 2 edges, and 3 vertices) in $N(x)$ amounts to a C_4 in $x \cup N(x)$. Moreover, denoting by $N(N(x))) = N^2(x))$ the set of neighbors of vertices in $N(x)$, it holds that $N(N(x))$ is also P_2-free, since the neighborhood of each vertex in $N(x)$ is P_2-free (considering what we stated above), and disjoint from one another (otherwise there would exist a C_4 in S). Naturally, the same holds for any $N^i(x)$.

Hence, in this case, the graph can be easily partitioned in 2 P_2-free subgraphs: starting from a arbitrary vertex that is colored black, one colors its neighborhood white, then the neighborhood of its neighborhood black, etc. Considering what we proved earlier, each color defines a P_2-free subgraph.

Finally, Hypothesis 1 is also verified when $k = 6$.

3 Reoptimization and BIN PACKING

Given a constant B, a list L of n items $L = (1, 2, \ldots, n)$, such that, for any $i = 1, \ldots, n$, its size $a_i \leqslant B$, and n bins each of capacity B, the BIN PACKING problem consists of arranging the items of L in the bins without exceeding their capacity (i.e., the sum of the sizes of the items placed in every bin must not exceed B) and in such a way that a minimum number of bins is used.

We show in this section that the basic technique used before in order to get inapproximability results can be also applied on hereditary problems not necessarily defined on graphs. This is, for instance the case of BIN PACKING. We will prove that this problem is inapproximable within approximation ratio $3/2 - \varepsilon$, for any $\varepsilon > 0$, in the reoptimization setting studied in this paper. For simplicity we consider a non-normalized instance of BIN PACKING where item i, $i \leqslant n$ has integer size a_i and bins have capacity B. We also assume that, for every i, $a_i < B$.

Suppose, ad contrario, that BIN PACKING is approximable within approximation ratio $3/2$ under items insertion and consider an instance I of the PARTITION problem, where $n + 1$ items $0, 1, \ldots, n$ with sizes $a_0, a_1, \ldots, a_n$ are given such that, for $i = 0, 1, \ldots, n$, $a_i < B$ and $\sum_{i=0}^{n} a_i = 2B$, and the objective is to find a partition of the items (if any) into two subsets such that the sum of the sizes of their items is equal to B. PARTITION is known to be **NP**-complete.

Assume now that items are ordered in decreasing size order (i.e., $a_0 \geqslant a_1 \geqslant \ldots \geqslant a_n$) and consider the list of items $L = (a_1, \ldots, a_n)$ as instance of BIN PACKING. Obviously, since $a_0 < B$ and $\sum_{i=0}^{n} a_i = 2B$, it holds that $\sum_{i=1}^{n} a_i > B$. Thus, an optimal BIN PACKING-solution for L has value greater than 1. We claim that L has a solution using 2 bins. Indeed, such a solution places the first k items in one bin, where k is the largest index such that $\sum_{i=1}^{k} a_i \leqslant B$, and the rest of the items from $k + 1$ to n in a second bin. Let us prove that such a solution is feasible, or equivalently, that $\sum_{i=k+1}^{n} \leqslant B$. Assume, ad contrario, that $\sum_{i=k+1}^{n} a_i > B$, recall that, by the definition of k, $\sum_{i=1}^{k+1} a_i > B$ and observe that $\sum_{i=0}^{k} a_i \geqslant \sum_{i=1}^{k+1} a_i > B$. In other words, on the hypothesis that $\sum_{i=k+1}^{n} a_i > B$, we derive that $\sum_{i=0}^{n} a_i = \sum_{i=0}^{k} a_i + \sum_{i=k+1}^{n} a_i > B + B = 2B$, a contradiction. So, the BIN PACKING-instance L has a solution of 2 bins. In all, we can assume that the initial BIN PACKING-instance is L and the optimal solution provided with is as just described.

Assume now that item 0 with size $a_0 \geqslant a_1$ arrives. On the hypothesis of the existence of a polynomial reoptimization algorithm achieving approximation ratio $3/2 - \varepsilon$ for BIN PACKING, if the instance I of PARTITION is a "yes"-instance, then this algorithm would provide a solution with 2 bins, while if I is a "no"-instance, the algorithm would provide a solution with at least 3 bins, deciding so in polynomial time PARTITION.

On the other hand, BIN PACKING is immediately approximable within $3/2$ in the reoptimization setting under consideration. Given an instance L and a solution with k bins, when an element arrives, one can open a new bin in order to place it (after, eventually, a quick check that all the items cannot be placed in

the same bin) guaranteeing so an approximation ratio $(k+1)/k \leqslant 3/2$, for $k \geqslant 2$, while the case $k = 1$ is trivially polynomial (just check whether $\sum_{i=1}^{n} a_i \leqslant B$, or not).

4 Conclusion

We have discussed the approximability of MAX WEIGHTED P_k-FREE SUBGRAPH in the reoptimization setting under vertex insertion. It appears that the initial optimum (or a good initial solution) provides a very useful information when approximating the modified instances, and we presented reoptimization algorithms which take advantage of this information in the best possible way, since the approximation ratio provided is the best constant ratio achievable in polynomial time (unless $\mathbf{P} = \mathbf{NP}$). In this paper, we have proved optimal results for MAX WEIGHTED P_k-FREE SUBGRAPH even if there exist 2 different weights 1 and M in the instance. We can easily prove that MAX WEIGHTED P_k-FREE SUBGRAPH has a PTAS for any fixed k. Moreover, we can adapt the proofs given in the paper to produce results depending on parameter M as done in [7]. Finally, in [11], the authors prove that the knowledge of all optimal solutions for free doesn't help TSP reoptimization. It is interesting to handle this question for MAX WEIGHTED P_k-FREE SUBGRAPH.

References

1. Archetti, C., Bertazzi, L., Speranza, M.: Reoptimizing the traveling salesman problem. Networks 42(3), 154–159 (2003), http://dblp.uni-trier.de/db/journals/networks/networks42.html\#ArchettiBS03
2. Archetti, C., Bertazzi, L., Speranza, M.: Reoptimizing the 0-1 knapsack problem. Discrete Applied Mathematics 158(17), 1879–1887 (2010), http://dblp.uni-trier.de/db/journals/dam/dam158.html\#ArchettiBS10
3. Ausiello, G., Bonifaci, V., Escoffier, B.: Complexity and approximation in reoptimization. CiE 2007: Logic and Computation and Logic in the Real World (2007)
4. Ausiello, G., Escoffier, B., Monnot, J., Paschos, V.: Reoptimization of minimum and maximum traveling salesman's tours. In: Arge, L., Freivalds, R. (eds.) SWAT. Lecture Notes in Computer Science, vol. 4059, pp. 196–207. Springer (2006), http://dblp.uni-trier.de/db/conf/swat/swat2006.html\#AusielloEMP06
5. Bilò, D., Böckenhauer, H.J., Hromkovic, J., Královic, R., Mömke, T., Widmayer, P., Zych, A.: Reoptimization of steiner trees. In: Gudmundsson, J. (ed.) SWAT. Lecture Notes in Computer Science, vol. 5124, pp. 258–269. Springer (2008), http://dblp.uni-trier.de/db/conf/swat/swat2008.html\#BiloBHKMWZ08
6. Bilò, D., Böckenhauer, H.J., Komm, D., Královi, R., Mömke, T., Seibert, S., Zych, A.: Reoptimization of the shortest common superstring problem. In: Kucherov, G., Ukkonen, E. (eds.) Proc. Combinatorial Pattern Matching, CPM'09. Lecture Notes in Computer Science, vol. 5577, pp. 78–91. Springer-Verlag (2009)
7. Bilò, D., Widmayer, P., Zych, A.: Reoptimization of weighted graph and covering problems. In: Bampis, E., Skutella, M. (eds.) WAOA. Lecture Notes in Computer Science, vol. 5426, pp. 201–213. Springer (2008), http://dblp.uni-trier.de/db/conf/waoa/waoa2008.html\#BiloWZ08

8. Böckenhauer, H.J., Forlizzi, L., Hromkovic, J., Kneis, J., Kupke, J., Proietti, G., Widmayer, P.: On the approximability of tsp on local modifications of optimally solved instances. Algorithmic Operations Research 2(2), 83–93 (2007), http://dblp.uni-trier.de/db/journals/aor/aor2.html\#BockenhauerFHKKPW07
9. Böckenhauer, H.J., Hromkovic, J., Královic, R., Mömke, T., Rossmanith, P.: Reoptimization of steiner trees: Changing the terminal set. Theor. Comput. Sci. 410(36), 3428–3435 (2009), http://dblp.uni-trier.de/db/journals/tcs/tcs410.html\#BockenhauerHKMR09
10. Böckenhauer, H.J., Hromkovic, J., Mömke, T., Widmayer, P.: On the hardness of reoptimization. In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM. Lecture Notes in Computer Science, vol. 4910, pp. 50–65. Springer (2008), http://dblp.uni-trier.de/db/conf/sofsem/sofsem2008.html\#BockenhauerHMW08
11. Böckenhauer, H.J., Hromkovič, J., Sprock, A.: Knowing all optimal solutions does not help for TSP reoptimization. In: Kelemen, J., Kelemenova, A. (eds.) Computation, Cooperation, and Life: Essays Dedicated to Gheorghe Paun on the Occasion of His 60th Birthday, Lecture Notes in Computer Science, vol. 6610, pp. 7–15. Springer-Verlag (2011)
12. Böckenhauer, H.J., Komm, D.: Reoptimization of the metric deadline TSP. J. Discrete Algorithms 8(1), 87–100 (2010), http://dblp.uni-trier.de/db/journals/jda/jda8.html\#BockenhauerK10
13. Boria, N., Monnot, J., Paschos, V.T.: Reoptimization of maximum weight induced hereditary subgraph problems. Cahier du LAMSADE 311, LAMSADE, UniversitÃl' Paris-Dauphine (Juin 2001)
14. Boria, N., Paschos, V.: Fast reoptimization for the minimum spanning tree problem. Journal of Discrete Algorithms 8(3), 296–310 (2010)
15. Escoffier, B., Milanic, M., Paschos, V.: Simple and fast reoptimizations for the steiner tree problem. Algorithmic Operations Research 4(2), 86–94 (2009), http://dblp.uni-trier.de/db/journals/aor/aor4.html\#EscoffierMP09
16. Lund, C., Yannakakis, M.: The approximation of maximum subgraph problems. Automata, Languages and Programming pp. 40–51 (1993)
17. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. In: Proc. STOC'06. pp. 681–690 (2006)